CC3301 Programación de Software de Sistemas

Control 1 – Semestre Primavera 2013 – Prof.: Luis Mateu

Pregunta 1

Se define el *formato decimal* como una representación de números positivos de gran tamaño en base decimal por medio de secuencias de bytes terminadas en 0xFF. Los siguientes ejemplos muestran este formato en acción:

número (en base 10)	*secuencia de bytes en hexadecimal*						
	0	*1*	*2*	*3*	*4*	*5*	*6*
0	FF						
1	10	FF					
1354	45	31	FF				
803312992103	30	12	99	21	33	08	FF

Observe que cada byte permite representar 2 dígitos decimales de un número y que el orden de almacenamiento en la secuencia es de menor a mayor significancia. Por lo tanto los 4 bits de *mayor* significancia en cada byte almacenan el dígito de menor significancia de ese byte (y viceversa). No hay límite para el tamaño que pueden alcanzar estos números.

Programe las siguientes funciones:

`int largo_dec(char *n);`	largo en bytes de n
`int extraer(char b, int n);`	extrae el dígito `n` de `b`
`int sum_bytes(char n1, char n2, int acarreo, char *r);`	suma 2 números de 2 dígitos c/u
`char *sum_dec(char *n1, char *n2);`	suma de números

La primera función calcula el número de bytes ocupados por un número.

En `extraer`, si n es 1, se entrega el dígito de menor significancia, y si n es 0, se entrega el dígito de mayor significancia. Por ejemplo `extraer(0x57,1)` retorna 5 y `extraer(0x57,0)` retorna 7.

La función `sum_bytes` suma los 2 números de 2 dígitos contenidos en `n1` y `n2` y un `acarreo` que puede ser 0 o 1 dejando el resultado en `*r`. Además retorna 1 si el resultado no era representable en un solo byte, o 0 en caso contrario. Por ejemplo `sum_bytes(0x48,0x34,1,&b)` debe dejar b en 0x82 y retornar 1.

Por último, la función `sum_dec` recibe 2 números representados en formato decimal y entrega una nueva secuencia que representa la suma de ambos números en el formato decimal. La nueva secuencia debe ocupar el mínimo número de bytes que permita representar el resultado de la suma.

Ejemplos de uso de `sum_dec`:

código	*formato decimal*	*valor*
`char n1[]= {0x10, 0xff};`	`10 FF`	`1`
`char n4[]= {0x40, 0xff};`	`40 FF`	`4`
`char n159[]= {0x95, 0x10, 0xff};`	`95 10 FF`	`159`
`char n9999[]= {0x99, 0x99, 0xff};`	`99 99 FF`	`9999`
`char *r1= sum_dec(n1, n4);`	`50 FF`	`5`
`char *r2= sum_dec(n4, n159);`	`36 10 FF`	`163`
`char *r3= sum_dec(n1, n9999);`	`00 00 10 FF`	`10000`

Restricción: Ud. *no puede usar* el operador de subindicación `[]` ni los operadores *binarios* `*`, `/`, `%` en ninguna de las funciones.

Pregunta 2

Un árbol binario se representa mediante la siguiente estructura:

```
typedef struct nodo {
  int x;
  struct nodo *izq, *der;
} Nodo;
```

Parte a.- (2 puntos) Programe una función que cuente los nodos que están a nivel n. El encabezado de la función es:

```
int contar_nivel(Nodo *A, int n);
```

Por ejemplo la raíz del árbol A está a nivel 0, sus hijos a nivel 1, sus nietos a nivel 2, etc.

Parte b.- (4 puntos) Programe una función que pode un árbol a partir del nivel n. El encabezado de la función es:

```
int podar_nivel(Nodo **A, int n, Nodo **cortados);
```

Esta función poda el árbol A cortando todos los subárboles que se encuentran a nivel n. Esta función también recibe como argumento la dirección de un arreglo de punteros en donde se deben dejar todos los subárboles que sean cortados. Por ejemplo, si A contenía 3 nodos a nivel 2, el arreglo cortados contendrá los 3 subárboles que fueron cortados y la función retornará 3. *Observe que podar el nivel 0 requiere modificar *A*.

Ejemplo:

```
Nodo *c[3];
int l= podar_nivel(&a, 2, c);
```